midwest architecture community collaboration

Data Lakes, Data Hubs and AI

Dan McCreary

Distinguished Engineer in Artificial Intelligence

Optum Advanced Applied Technologies

AEA Association of Enterprise Architects Minnesota Chapter

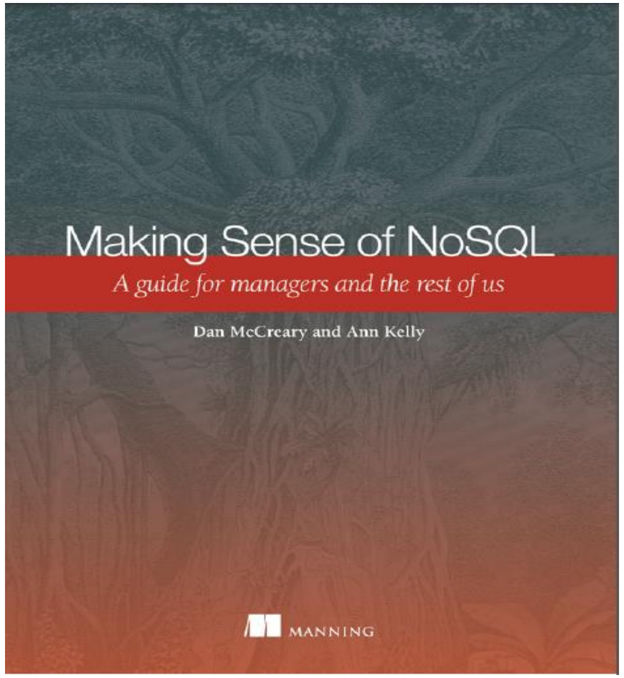
Enterprise Architecture Partnership Talk Twin Cities

IASA

Twin Cities Enterprise Architecture Roundtable

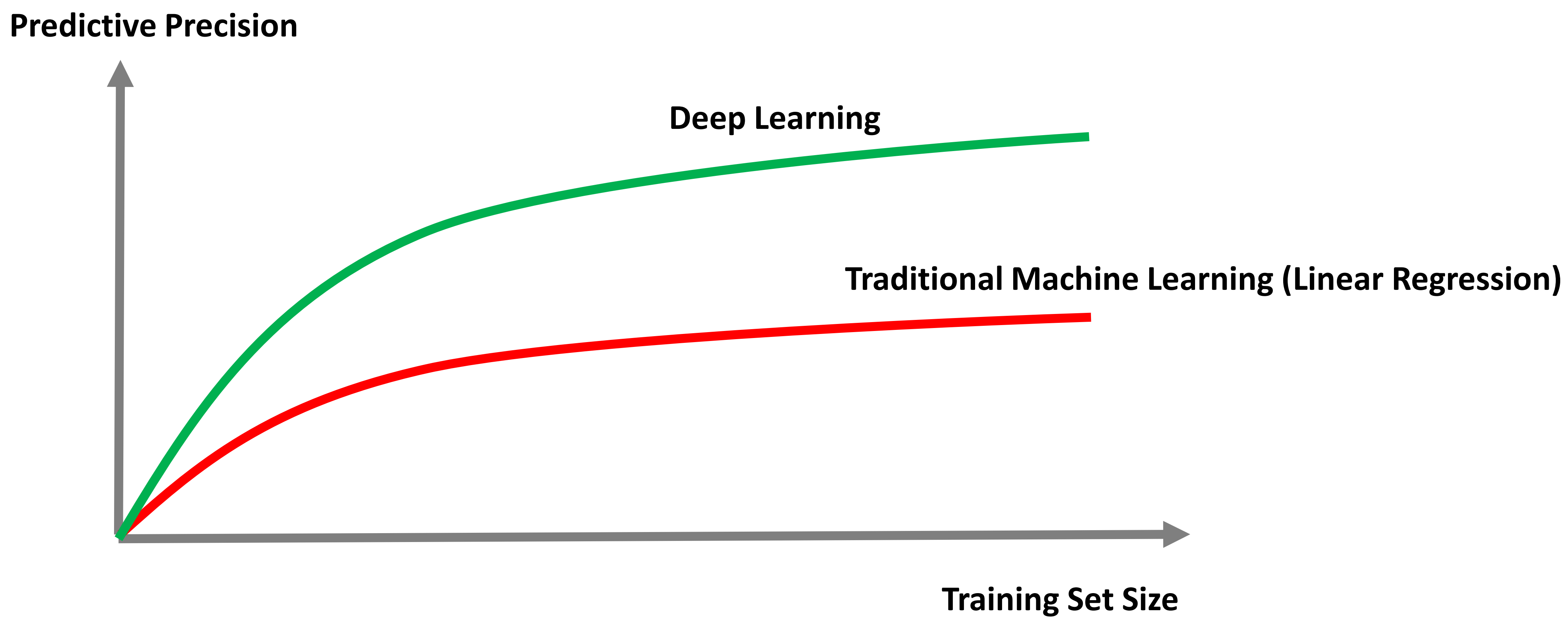
Background for Dan McCreary

- Co-founder of "NoSQL Now!" conference
- Coauthor (with Ann Kelly) of "Making Sense of NoSQL"
    - Guide for managers and architects
    - Focus on NoSQL architectural **tradeoff** analysis
    - Basis for **40 hour course** on database architectures
    - How to pick the right database **architecture**
    - http://manning.com/mccreary
- Focus on metadata and IT strategy (capabilities)
- Currently focused on NLP and Artificial Intelligence Training Data Management



midwest architecture community collaboration

The Impact of Deep Learning



Large datasets create competitive advantage

midwest architecture community collaboration

High Costs of Data Sourcing for Deep Learning

80 % OF TIME **WASTED**

By data scientists just getting access to data and preparing data for analysis

60 % OF THE **COST**

Of data warehouse projects is on ETL

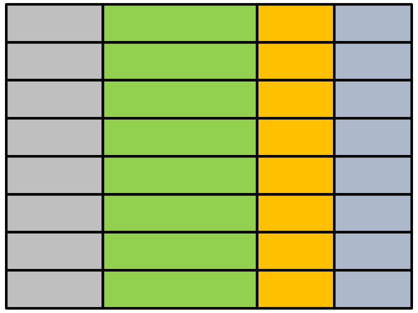
$3.5 BILLION IN **SPENDING**

In 2016 on data integration software

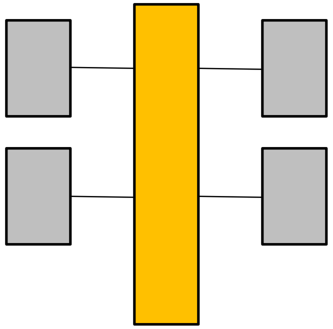
midwest
architecture community
collaboration

Six Database Core Architecture Patterns

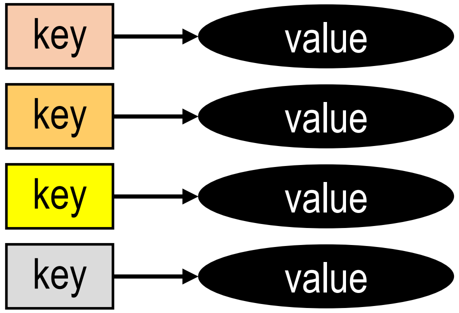
Relational



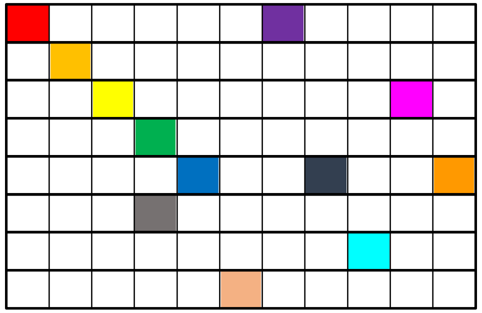
Analytical (read-mostly OLAP)



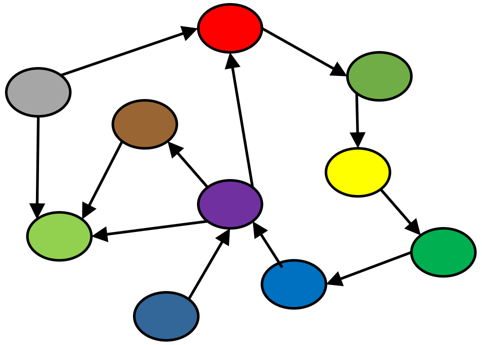
Key-Value



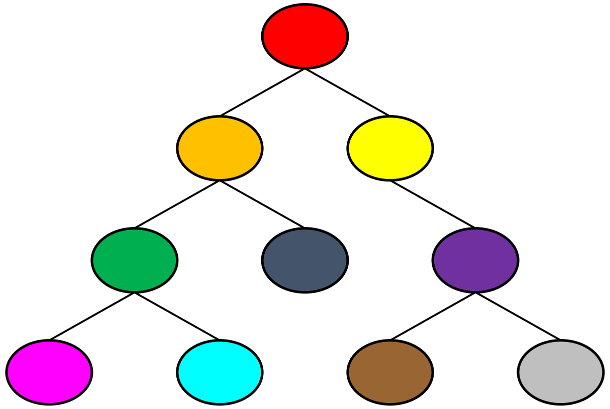
Column-Family



Graph



Document



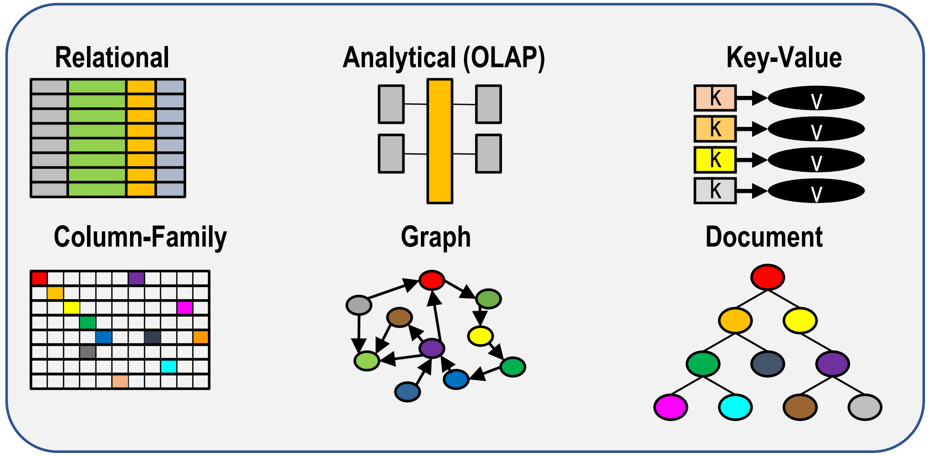
midwest architecture community collaboration

Which architectures are best for data lakes, data hubs and AI?

Role of the Solution Architect

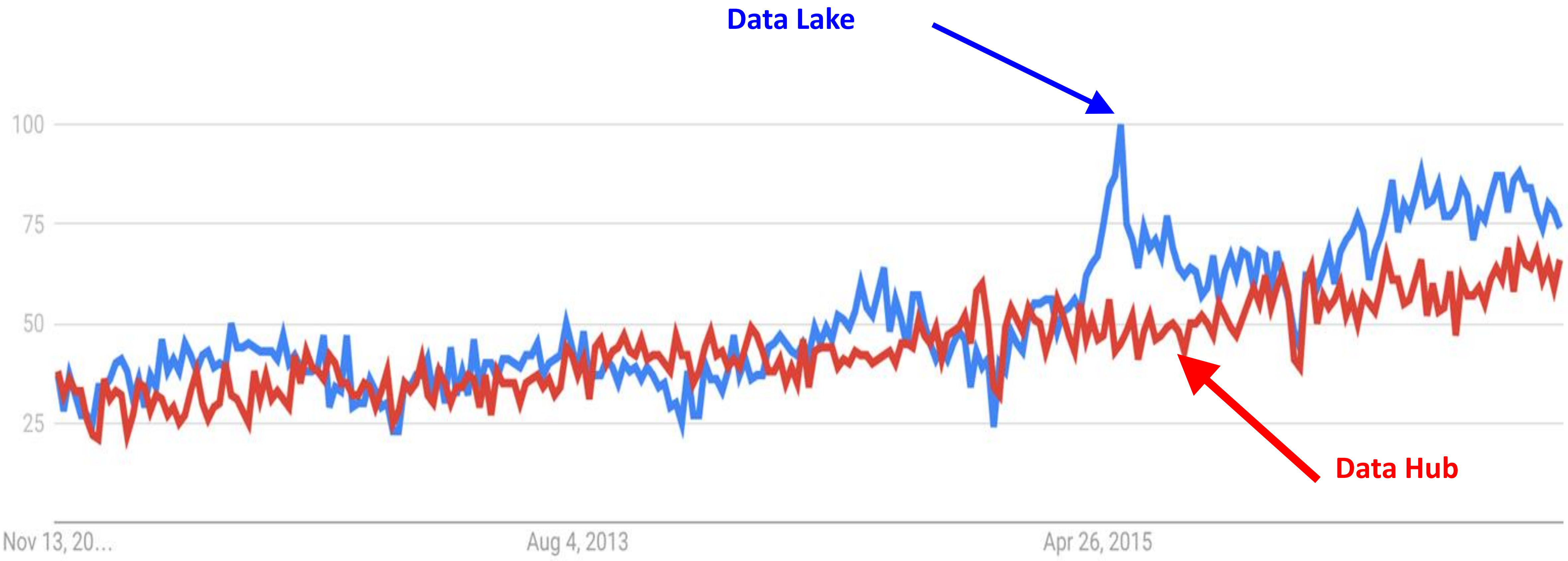
Business Unit

Sally Solutions
Title: Solution Architect

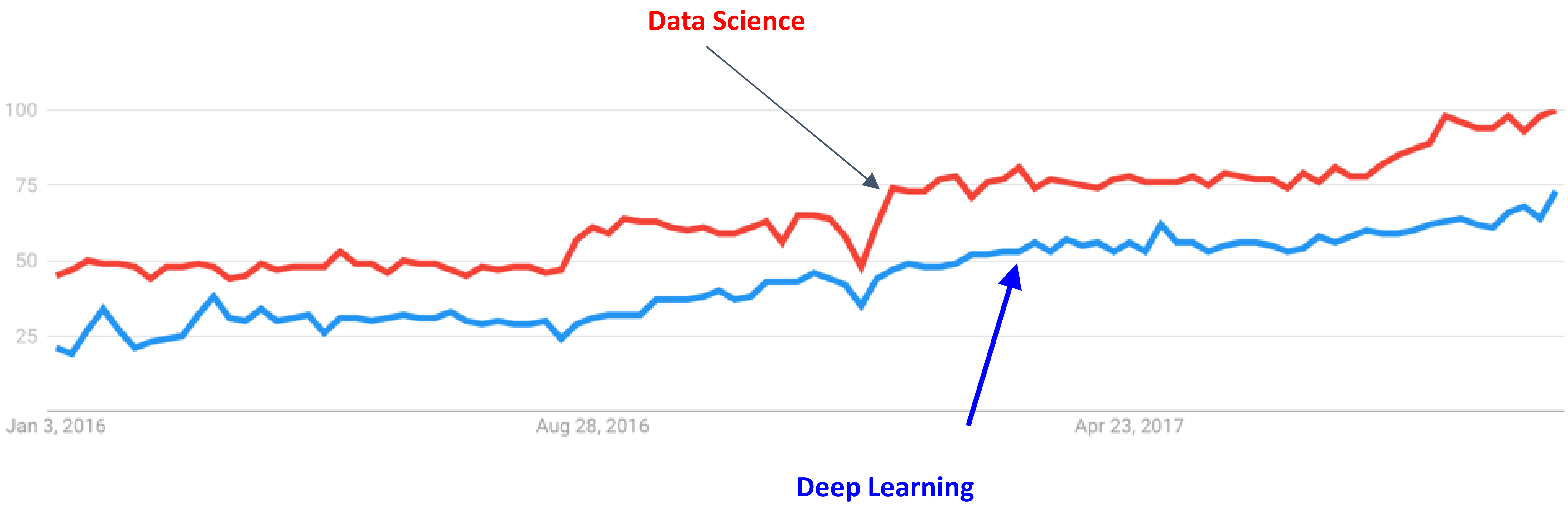


- Non-bias matching of business problem to the right data **architecture** **before** we begin looking at a specific **products**

Google Trends



Data Science and Deep Learning



Data Lake Definition

~10 TB and up
$350/10TB Drive

*A storage repository that holds a **vast** amount of **raw** data in its **native format** until it is needed.*

Examples of raw native format:

- Dump of data from an RDBMS in csv format
- Export data with many numeric codes
- Log files

http://searchaws.techtarget.com/definition/data-lake

midwest architecture community collaboration

Data Lake Assumptions

- Scale-out architecture
  - Adding more data won't slow down reads
  - No "joins" are ever used to access data
- Consistency
  - Consistent read and write performance, even under heavy load
- High availability and tunable replication
  - Default replication factor = 3
- No secondary indexes
- Low cost
  - $500/TB/year (Amazon S3 is at under $360/TB/year)

Amazon S3 Pricing (Nov. 2016)

23 cents/GB/year = $230/TB/year

Region: US East (Ohio)

	Standard Storage	Standard - Infrequent Access Storage †	Glacier Storage
First 50 TB / month	$0.023 per GB	$0.0125 per GB	$0.004 per GB
Next 450 TB / month	$0.022 per GB	$0.0125 per GB	$0.004 per GB
Over 500 TB / month	$0.021 per GB	$0.0125 per GB	$0.004 per GB

Service Level Agreement (SLA) Design:

- 5 “9s” availability (you can read your data at any point in time)
- 11 “9s” durability (your data will not get lost)
- Price has **never** gone up (only down)

midwest architecture community collaboration

Implemented with a distributed fault-tolerant parallel file systems

- Hadoop
  - Hadoop Distributed File System – HDFS
  - Default replication level 3
  - 128MB block size designed for write once, read may
- Amazon S3
  - Cloud based object store – cost leader for High Availability
- Other Distributed File Systems:
  - Ceph (OpenStack)
  - GlusterFS (now owned by Red Hat)
  - GNU Cluster File System
  - Lustre

https://en.wikipedia.org/wiki/List_of_file_systems#Distributed_parallel_fault-tolerant_file_systems

Data Hub Definition

Meaningful to data subscribers

*A collection of data from multiple sources organized for **distribution**, **sharing**, and **subsetting**.*

You can query it!

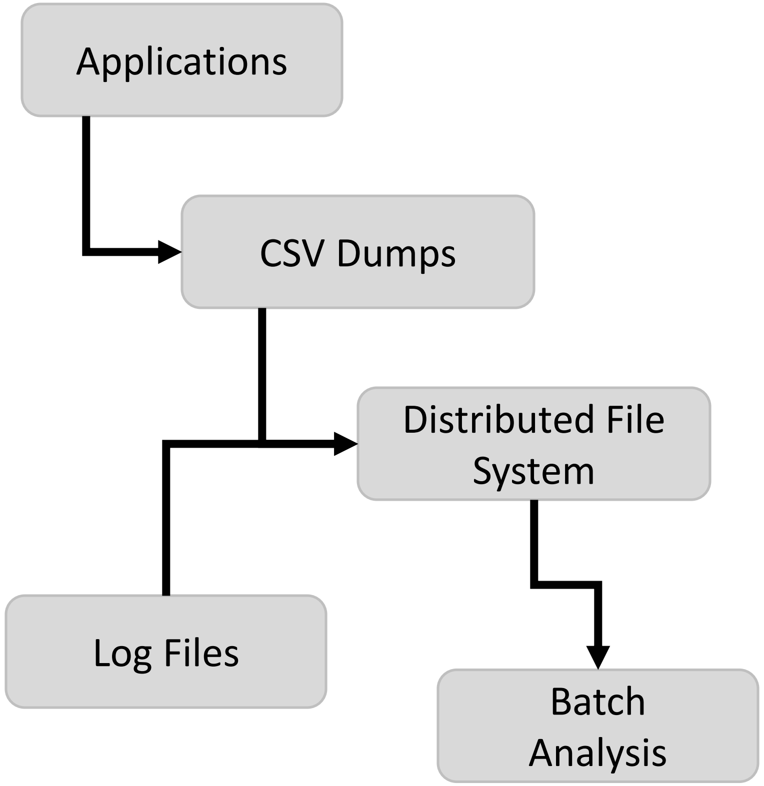
*Generally this data distribution is in the form of a **hub and spoke** architecture.*

A data hub differs from a data lake by homogenizing data and possibly serving data in multiple desired formats, rather than simply storing it in one place, and by adding other value to the data such as de-duplication, quality, security, and a standardized set of query services.

https://en.wikipedia.org/wiki/Data_hub (Nov. 2016)

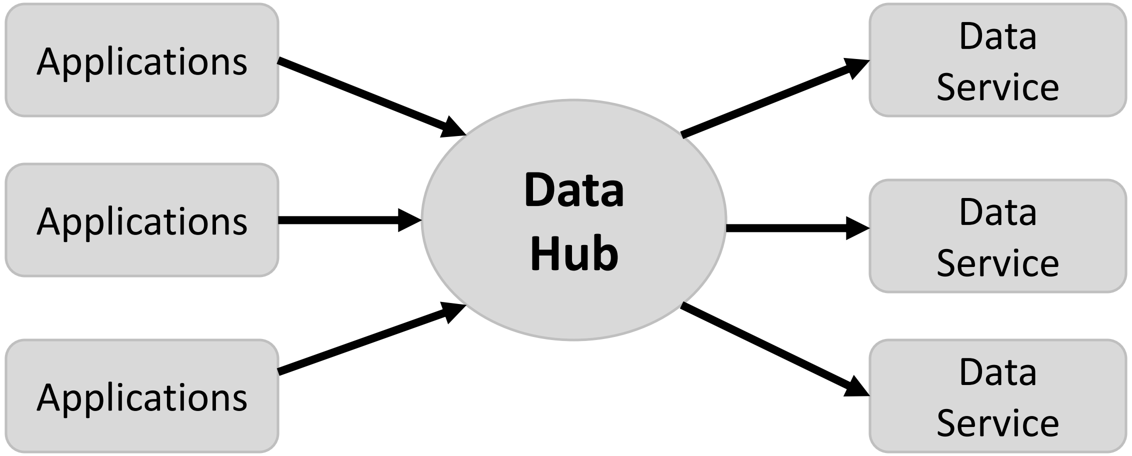
Data Flow Comparison

Data Lake



Not real-time

Data Hub

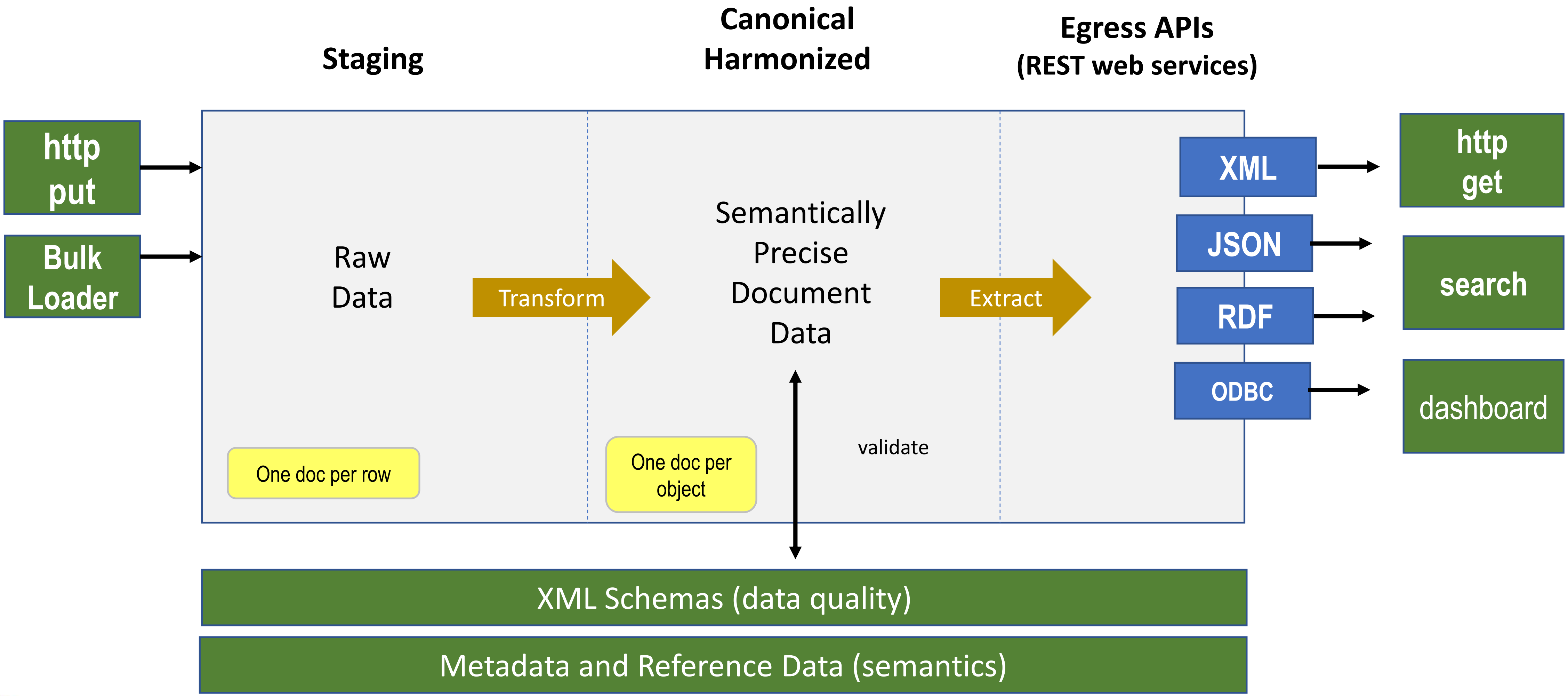


Sub-millisecond response times

Which one generates web pages?

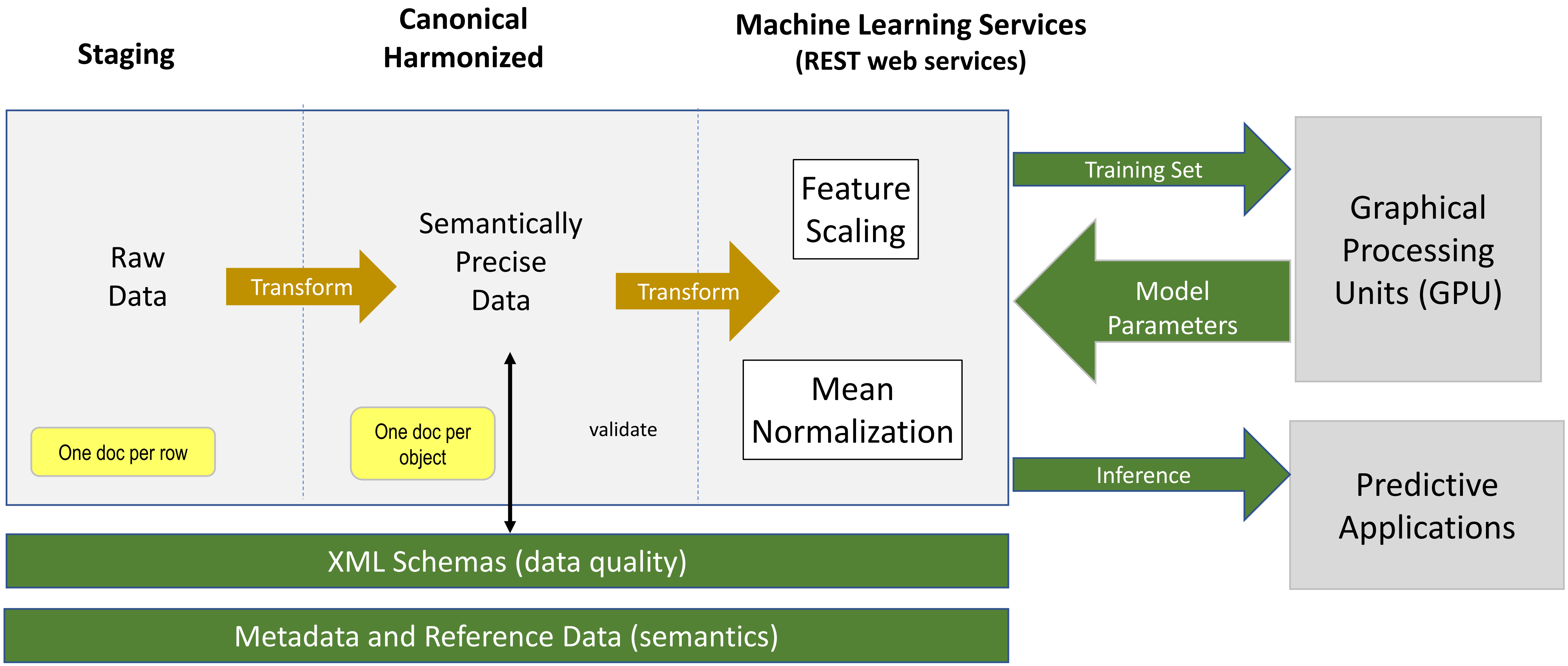
midwest architecture community collaboration

Sample Document Data Hub Data Flow Diagram



midwest architecture community collaboration

A.I. Driven Strategies



midwest architecture community collaboration

GAFA 2016 R&D Spending Amounts

1. **G**oogle - $16B
2. **A**mazon - $16B
3. **F**acebook - $6B
4. **A**pple - $10B

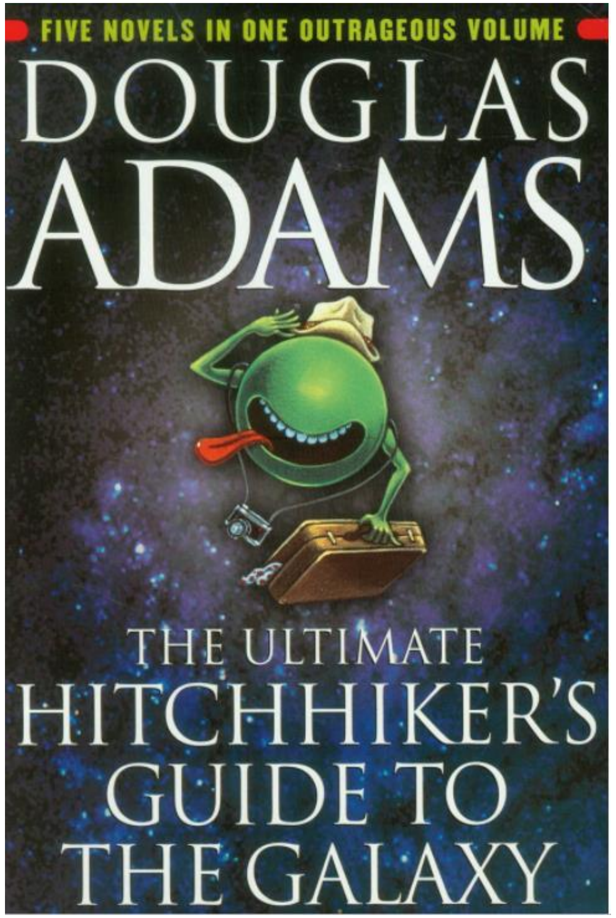
Total - $48 billion in annual R&D spending

https://www.recode.net/2017/9/1/16236506/tech-amazon-apple-gdp-spending-productivity

midwest architecture community collaboration

Or “GAFAM” and “GAFAMI” if you include Microsoft and IBM

How do data lakes answer the question…



What is the answer to life, the universe and everything?

<answer>42</answer>

midwest architecture community collaboration

Seven Levels of Semantics

Typical CSV data

What does this mean?

1 <INDGENCD>42</INDGENCD>

2 <Gender>42</Gender>

3 <PersonGenderCode>42</PersonGenderCode>

4 <PersonGenderCode>F</PersonGenderCode>

5 <PersonGenderCode>Female</PersonGenderCode>

6 <PersonGenderCode>Female</PersonGenderCode>

7 <c:PersonGenderCode>Female</c:PersonGenderCode>

xmlns:c="http://example.com/canonical-reference-data

RDF (next page)

Clear Meaning

Search Friendly!

Low Semantics

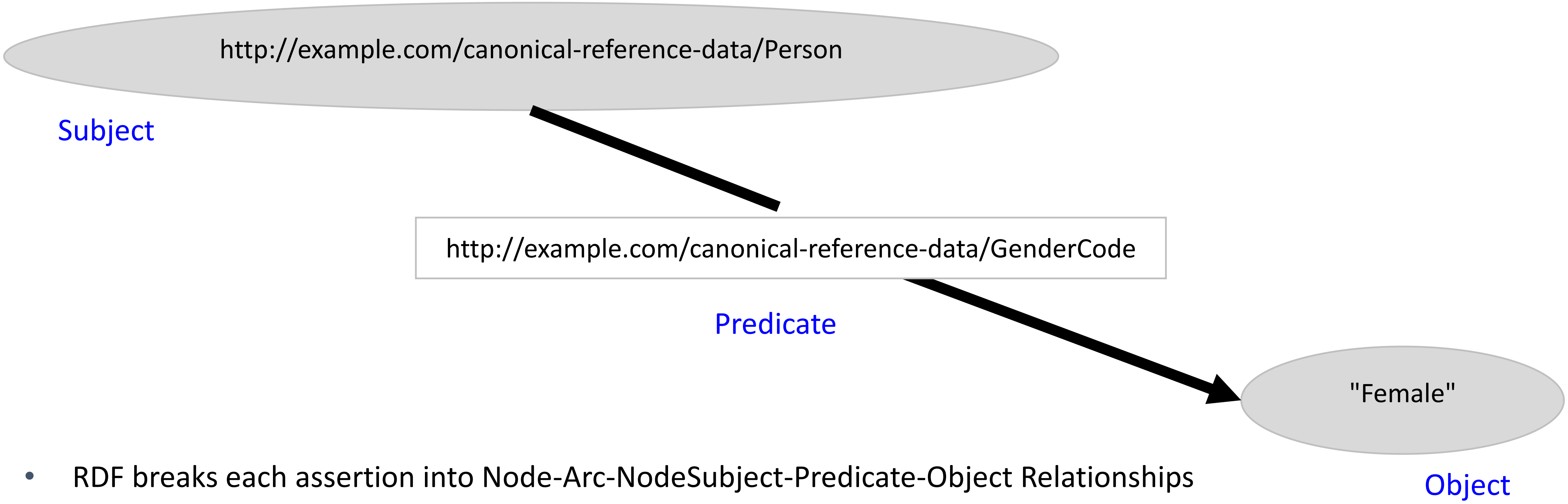
Low Data Governance

Data Stewards

High Semantics

midwest architecture community collaboration

Resource Description Format



- RDF breaks each assertion into Node-Arc-NodeSubject-Predicate-Object Relationships
- Subject and Predicates are URLs. Objects are sometimes "literal" strings
- If all imported data used RDF than transformation and integration would be trivial (no ETL)

midwest
architecture community
collaboration

The Semantic Spectrum

Low Semantics — High Semantics

Data Lake — Data Hub

1. Mostly Numeric Codes
2. No harmonization
3. Write and read by the source application
4. No namespace and validation
5. No data quality

1. Numbers **and** labels
2. Incremental harmonization
3. Writes and read by everyone
4. Validation
5. Data quality for all egress documents

Note that high-semantics are not "free"

It requires a large framework of tools to convert numeric codes into useful labels

It's About Building Integrated Views (360 views)

Integrated views of customers - every touchpoint visible by call center

Integrated views of hardware devices - every desktop, server, firewall etc.

Integrated views of whatever….

midwest
architecture community
collaboration

100% Failure Rate

The Old Way: Comprehensive Enterprise Modeling

First a brief word on the old approach. People used to (and occasionally still) build a new enterprise data model comprising every field and value in their existing enterprise, across all silos, and then map every silo to this new model in a new data warehouse via ETL jobs.

ITBusinessEdge surveyed companies and found that this approach **always** fails. Survey respondents report that it goes over budget or fails 100% of the time.

http://www.itbusinessedge.com/interviews/three-reasons-big-data-projects-fail.html

The Challenge of Data Variability in RDBMS Systems

“Our ER modeling process is taking too long.”

“Every time we think we have our ER model finalized there is another change request.”

“It takes us a long time to update or models after we have 100M rows of test data loaded.”

“These hidden one-to-many relations have slowed down our teams progress to a crawl.”

“We have no way to predict future data variability.”

“Exceptions make the rules. Each new system load has 10% variability.”

“Our system will be obsolete the day after we go to production.”

“Relational databases are like concrete – ones they set they are difficult to change.”

Perfection is the Enemy of Progress

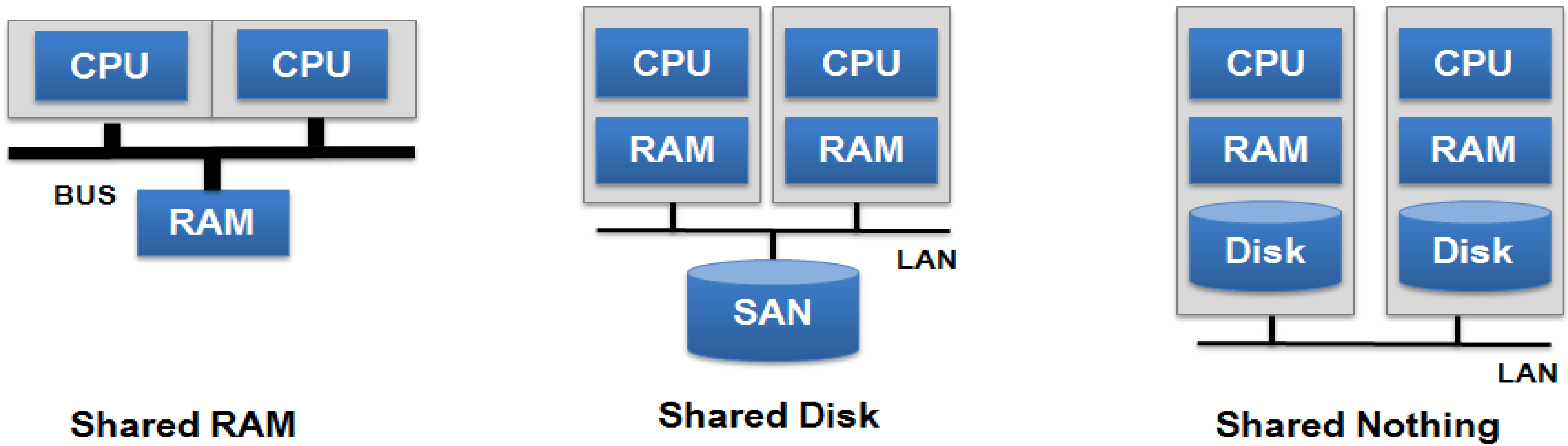
midwest architecture community collaboration

What Data Lakes and Data Hubs both have in common

1. They both are **NOT** relational (Yeah)!
2. No data modeling before you load your data! -> Agility, Flexibility (Schema Agnostic)
3. They both leverage low-cost shared-nothing commodity hardware
4. They both know how to reliably distribute computing loads over hundreds of processors
5. The both help organization understand the challenges of distributed computing

midwest
architecture community
collaboration

Low-Cost Scalability: Shared Nothing Architecture



Every node in the cluster has its own CPU, RAM and disk - but what about GPUs?

midwest architecture community collaboration

Fallacies of Distributed Computing

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

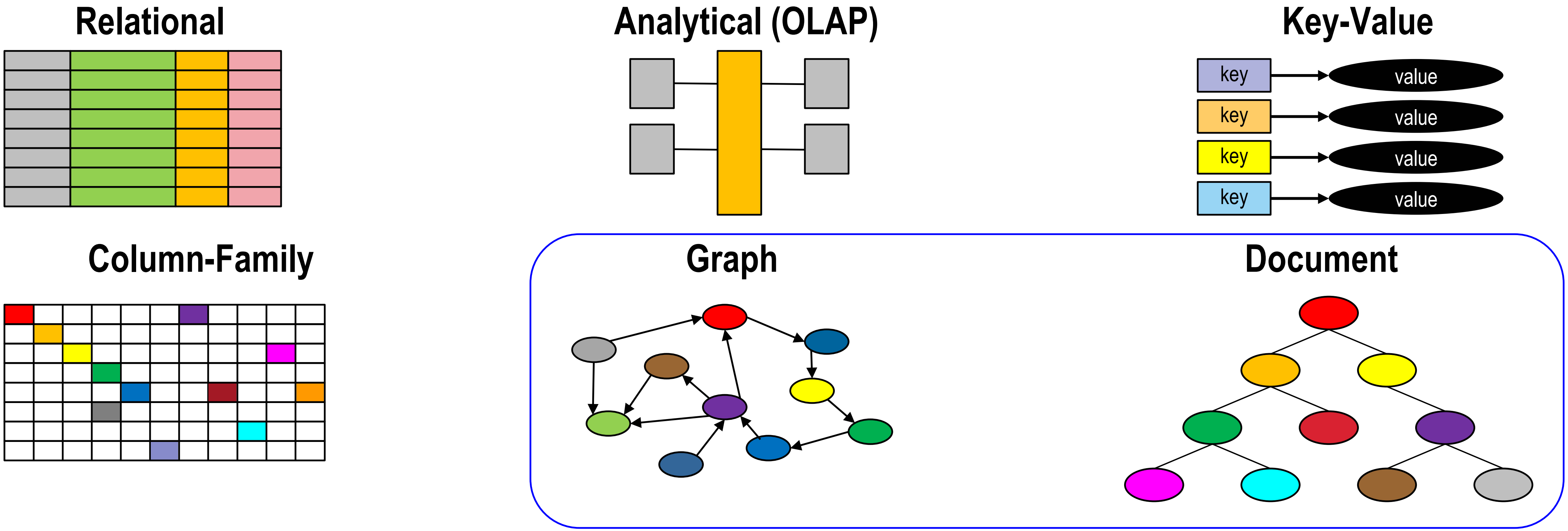
L Peter Deutsch
https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing

Data Hub Philosophy

- Ingest everything
- Index everything
- Analyze everything from the indexes
- Track data quality
- Incrementally harmonize
- Promote strong data governance and data stewardship
- Make it easy to do transactions, search and analytics on harmonized data

midwest
architecture community
collaboration

Document-Centric Data Hubs



- Document databases are **ideal** for many egress tasks
- Graphs make it easy to link related documents together

midwest architecture community collaboration

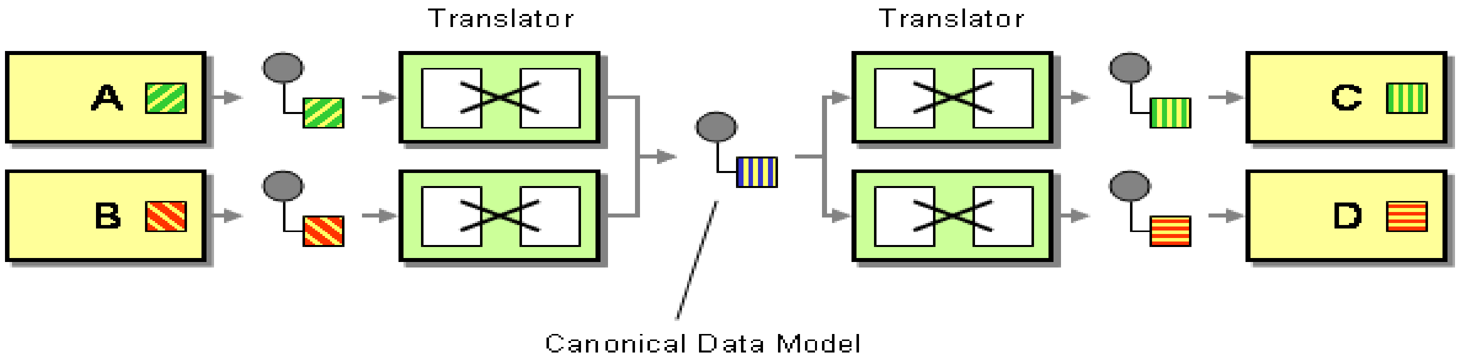
What is an Enterprise Canonical Model?

ca·non·i·cal
/kəˈnänək(ə)l/
adjective

1. according to or ordered by canon law.
   "the canonical rites of the Roman Church"
2. included in the list of sacred books officially accepted as genuine.
   "the canonical Gospels of the New Testament"

noun

1. the prescribed official dress of the clergy.
   "Cardinal Bea in full canonicals"



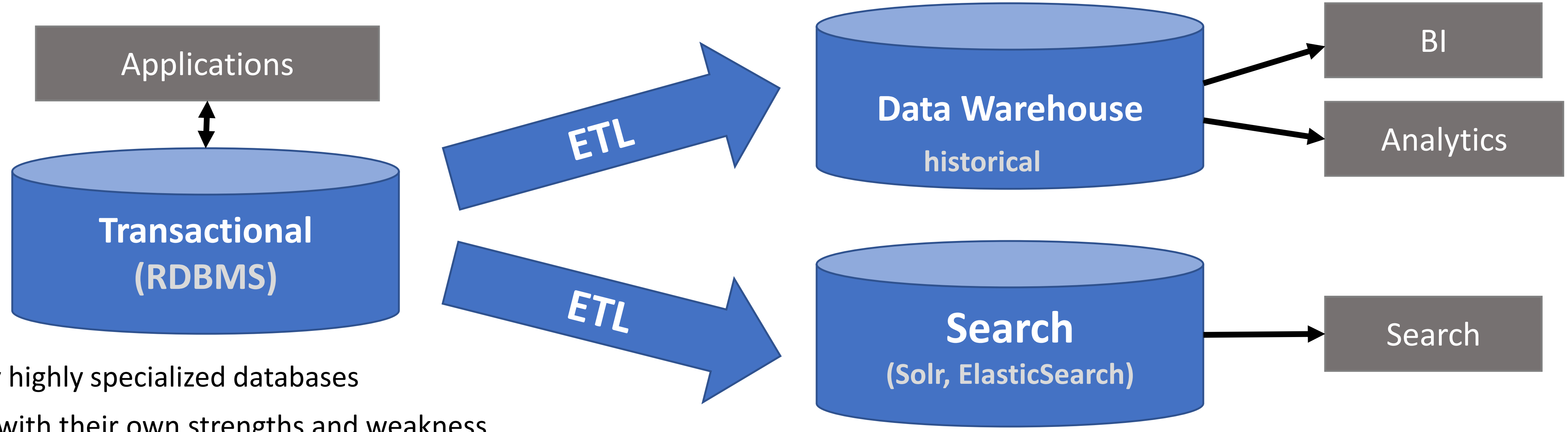
How can we minimize dependencies when integrating applications that use different data formats?

Design a *Canonical Data Model* that is independent from any specific application. Require each application to produce and consume messages in this common format.

http://www.enterpriseintegrationpatterns.com/CanonicalDataModel.html

midwest
architecture community
collaboration

Many-Component Specialized DB Cost Model



Many highly specialized databases

Each with their own strengths and weakness

Expensive ETL (batch and real-time) to keep databases in sync

Chargebacks based on CPUs and disk storage (not I/O)

Little understanding of the costs of moving data between systems

Total cost = RDBMS + ETL + DW + ETL + Search

Where do I store my metadata?

How can we eliminate the ETL?

midwest architecture community collaboration

Traditional EDW (and ODS) Pain Points

Non-trivial delivery time and effort

Dependency on Extract, Transform and Load (**ETL)** and **movement** of a lot of data

Very **brittle** with respect to change

Not suited for **unstructured** data

Legacy RDBMS technology **does not scale** flexibly or economically

midwest architecture community collaboration

Challenges with ETL

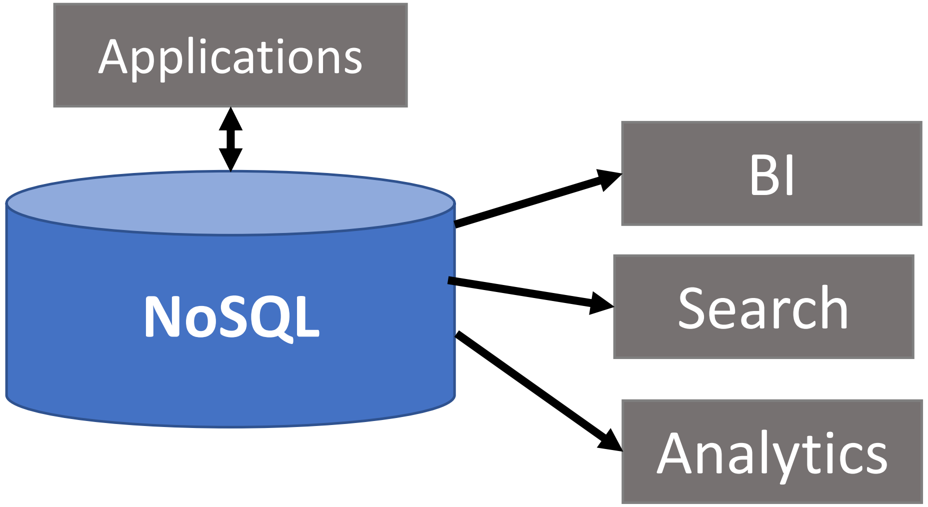
- Designed for table-to-table transformations
- Written in SQL with memory intensive "joins"
- Difficult to scale
- Difficult to re-use centralized business rules (no document validation)
- Batch transforms typically run over night
- Limited time windows
- Little chance for recovery on errors

midwest architecture community collaboration

Ideal NoSQL Cost Model: The Multi-Model Approach

Imagine a single database for:

- **All** transactions
- **All** Search
- **All** Analytics



Use a scale out architecture with ACID transactions

Index **everything** for fast queries, search and deep analytics

Avoid moving data around

Total costs can be much lower!

midwest architecture community collaboration

Why Use a Document Store?

Document stores are ideal when you have highly-variable data

- Example: clinical healthcare data

Document stores can be designed to have a "scale-out" horizontal-scalable architecture

- Unlike relational models, there are limited "join" operations
- Simply add new nodes to a "cluster" and the system can "rebalance" and support higher volumes of data

How are **Document-Centric** Data Hubs Different?

Handles complex data

- Does not fit well into a single table

Handles highly variable:

- Example: healthcare.gov
  - 37 states – 37 variations

Diverse users:

- Examples: Clinical, Claims, Analytics, Search, Research, Pharma

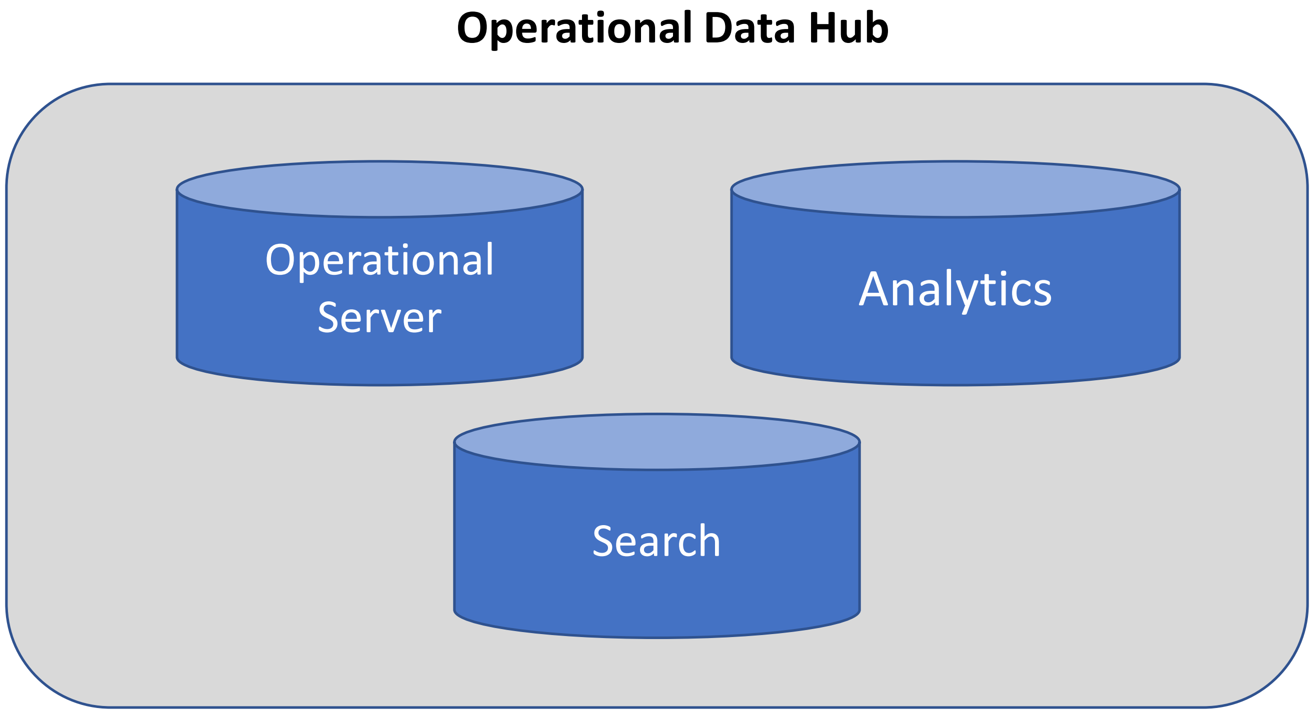
High security and audit (PHI)

- Requires role-based access control

Volume

- Requires a true scale-out architecture

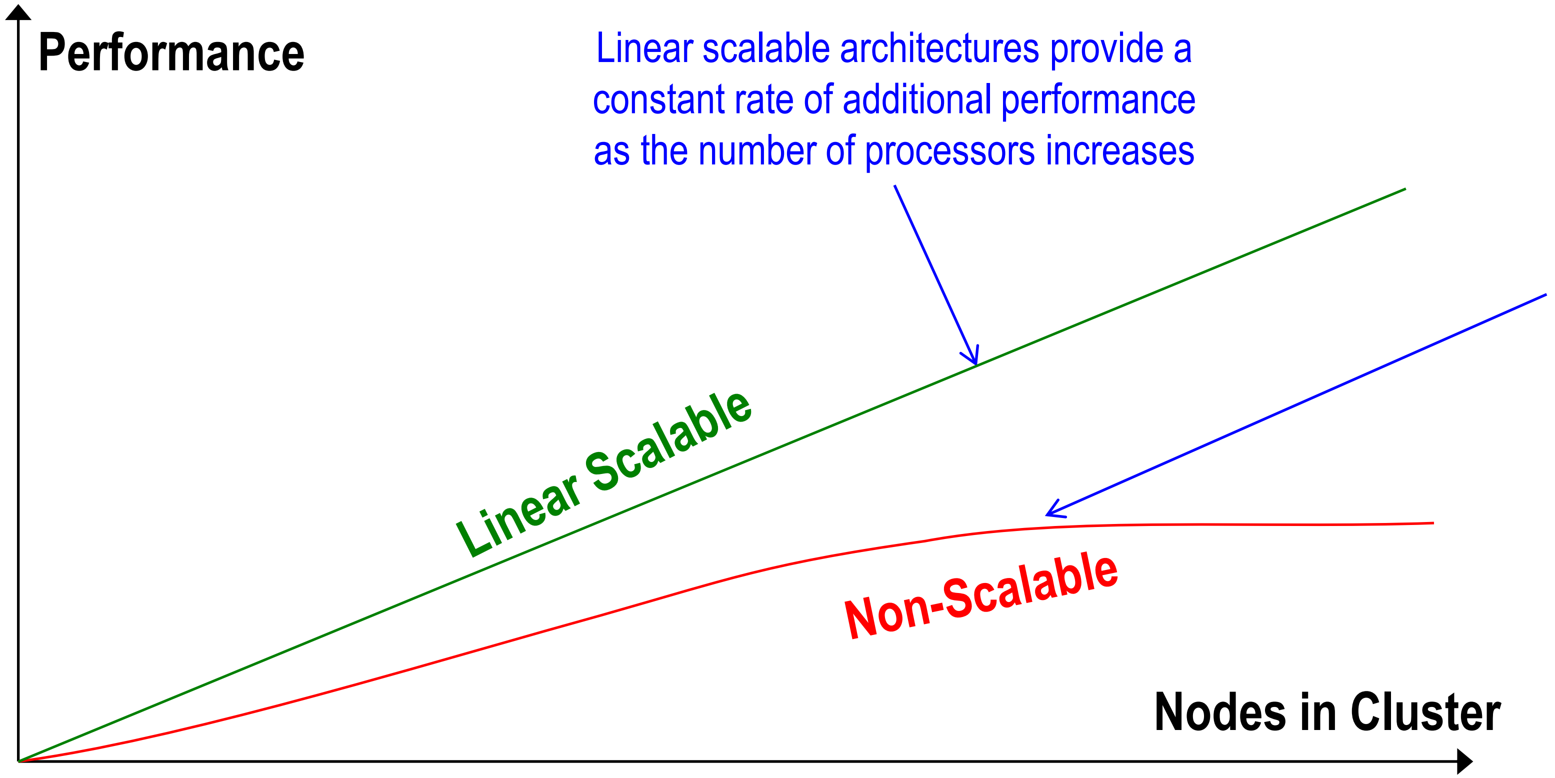
Three Functions – One Cluster – One Set of APIs



Combines a transaction safe application server, a database server and a search engine in the same server. Minimize data movement and ETL costs.

midwest architecture community collaboration

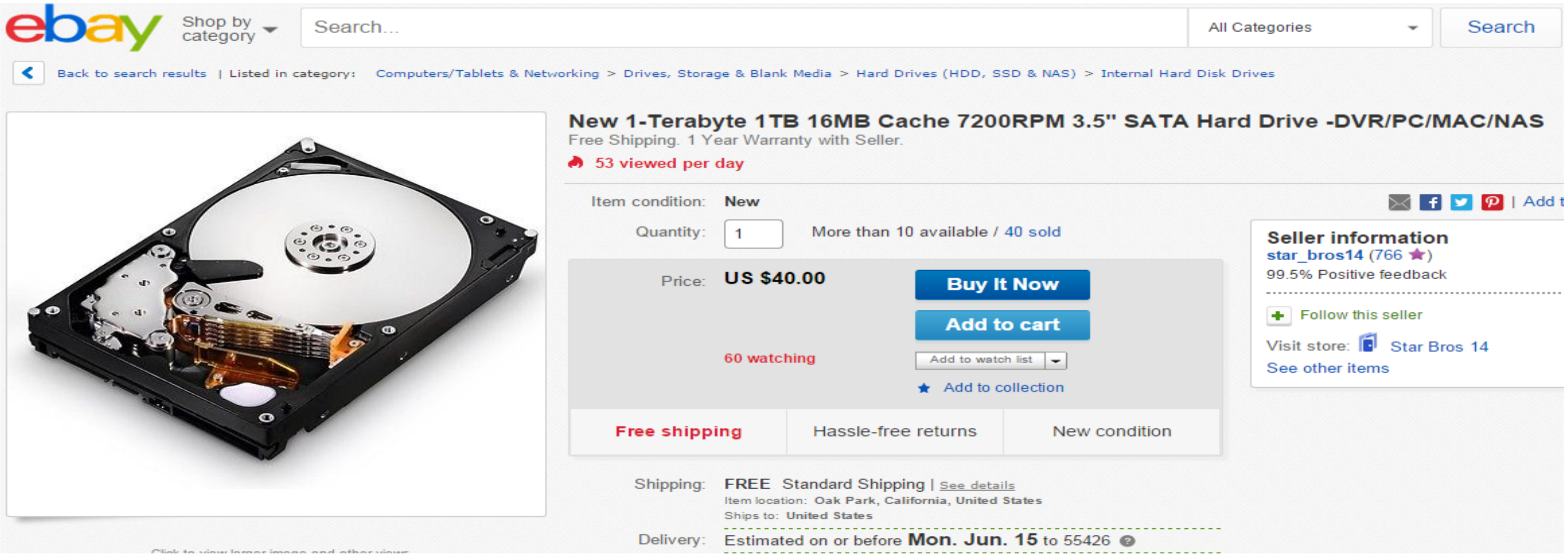
Horizontal Scalability



midwest architecture community collaboration

Data Storage Cost Models

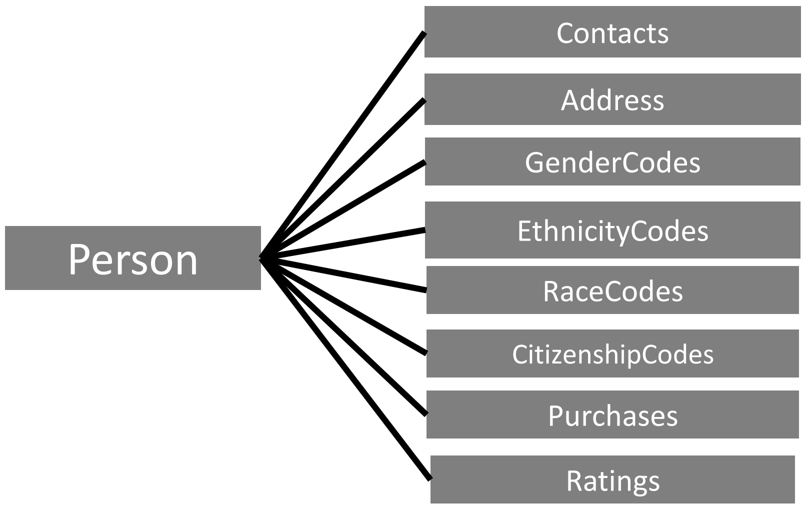
- Measured in dollars per TB per year ($/TB/Y)
  TB = Terabyte
- 1 TB hard drives cost = $40 (qty 1)



midwest architecture community collaboration

Comparison of Normal vs. Denormal Forms

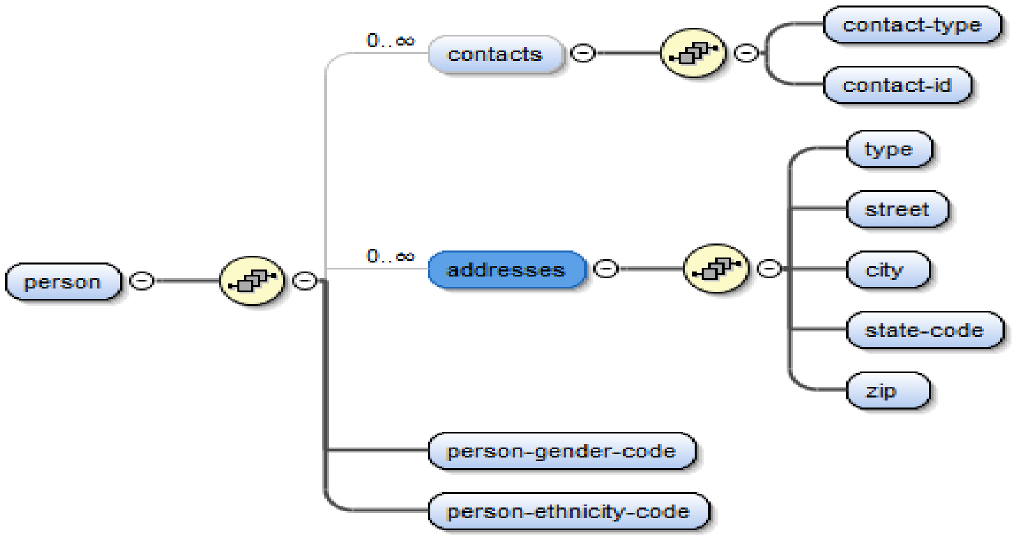
- Normalized



One join per child table

17 tables = 16 joins

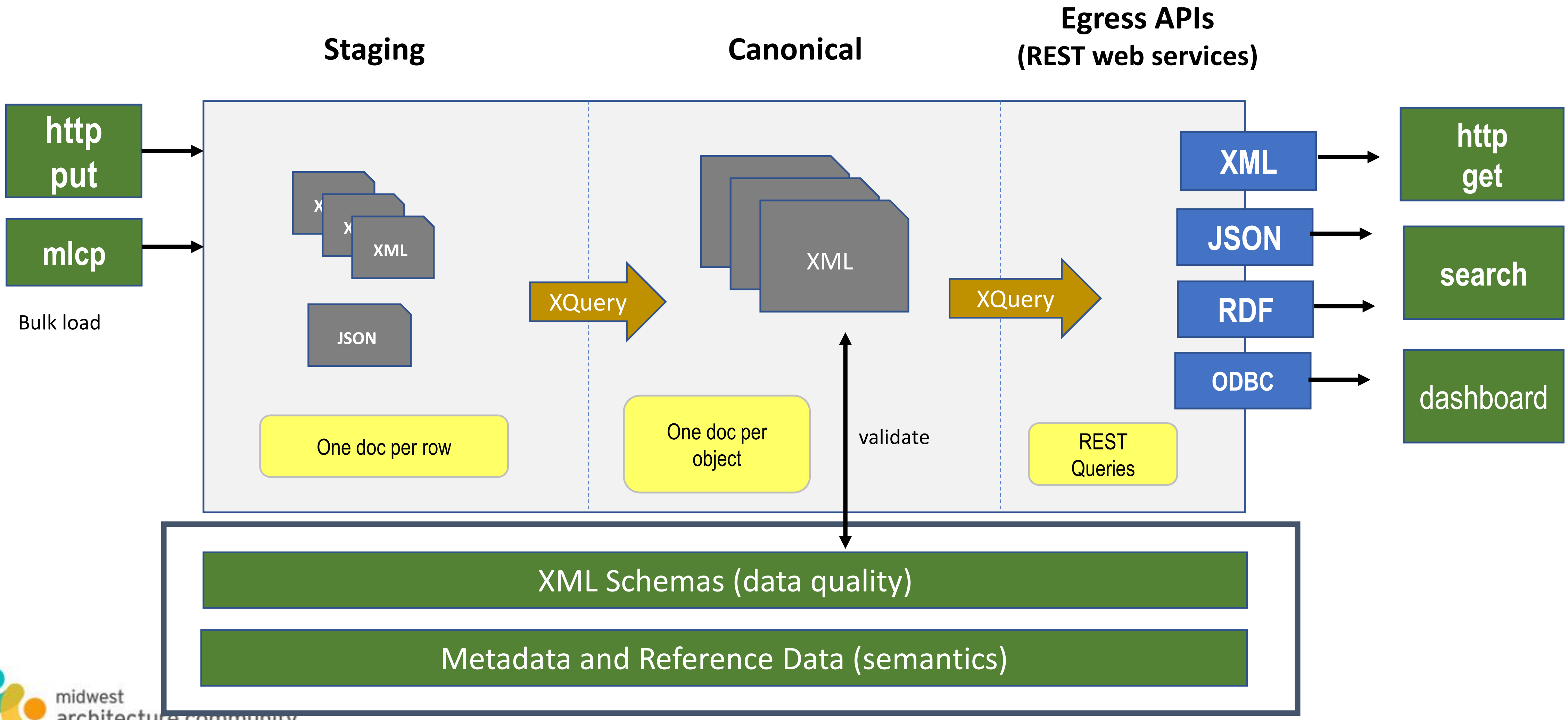
- Denormalized Document



- One document
- Single line of code: `get-person('123')`

midwest architecture community collaboration

Sample Data Hub Data Flow Diagram



midwest architecture community collaboration

Definitions

Staging

- Raw data
- One document per row from RDBMS
- Simple flat data

Canonical

- De-normalized and enriched data
- Semantically clear
- Validated by XML Schemas

Egress

- REST web services of high-quality data
- May include the use of pre-calculated aggregates (materialized views) to speed analytical reports
- Multiple formats
- Many options
- Strong SLAs (read times, high availability)

Reference Data

Data that is not associated with each new transaction

Data elements that use the "Code" suffix

- ISO-11179 Representation Term

Based on standards

- 80% of the tables in CDB is reference data

Goal:

- Consistent usage over time and between projects

Examples:

- Gender Code
- Ethnicity Code
- US State Code

Sample Reference Codes

user: dmccreary

Home

Reference Data Codes

Code Name	File Name	Record Count	Last Modified
citizenship-status-type	citizenship-status.xml	7	Thu, Apr 30 '15 15:21:31
gender-code	gender.xml	3	Thu, Apr 30 '15 13:35:57
marital-status	marital-status.xml	9	Thu, Apr 30 '15 11:45:06
race-type	race-type.xml	16	Thu, Apr 30 '15 13:52:57
state-abbr	state-abbr.xml	50	Mon, May 04 '15 20:55:59

Reference Code Mapping

Mapping Name	File Name	Record Count	Last Modified
marital-status-mapping	marital-status-mapping.xml	7	Thu, May 07 '15 10:35:57
race-type-mapping	race-type-mapping.xml	20	Thu, May 07 '15 10:36:12

Execution Time: 0.153703 seconds.

Back to Demo

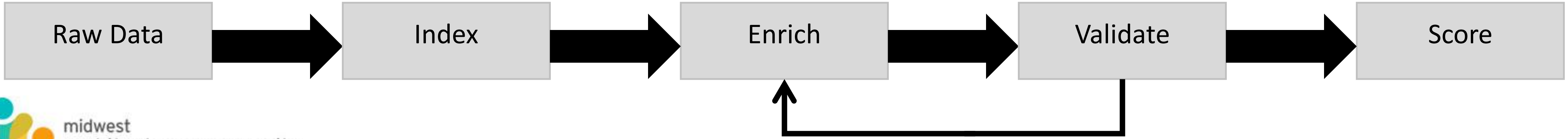
Data Management - C360 on MarkLogic
2015-05-14-05:00

How to build semantically useful systems?

- Muddy Data Lake

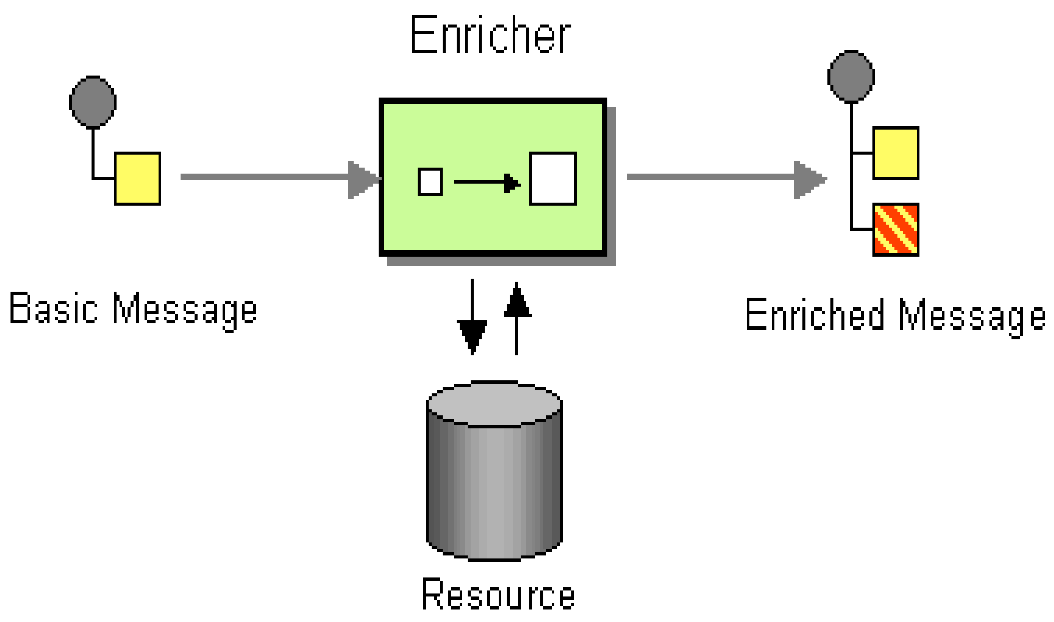
- Clear Data Hub

Build Continuous Enrichment Pipelines



midwest
architecture community
collaboration

Content Enrichment



Continuous process of enriching content using resources from your data hub

midwest architecture community collaboration

Reference Data Enrichment Services

- Conversion of low-semantics elements to high-semantics and search friendly forms
- Example:
  - Input: street, city, state
  - Output: add longitude and latitude

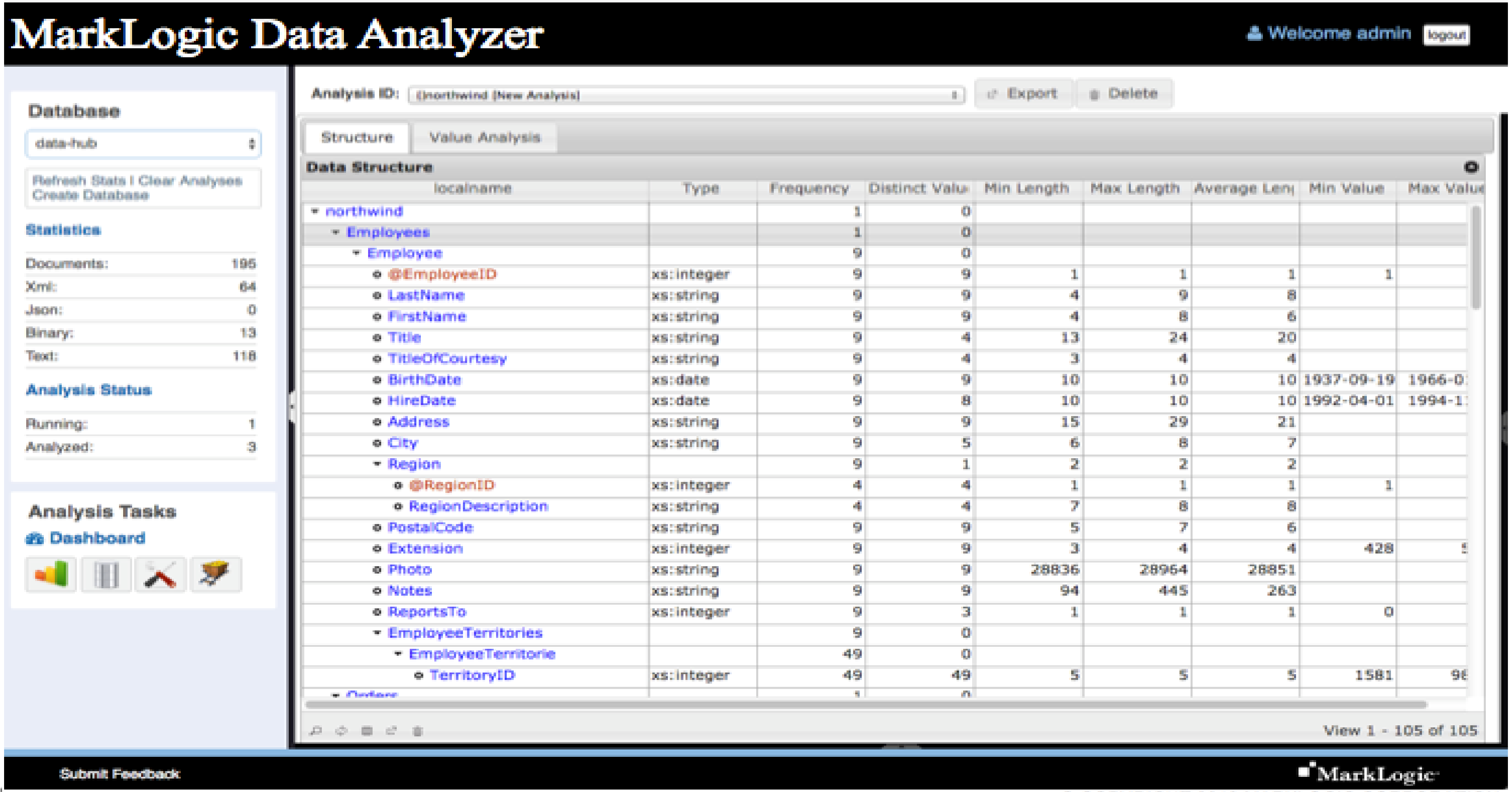
midwest architecture community collaboration

Data "Opaqueness"

- **Muddy**
  - No precise definitions and validation for data elements and code values
  - Expensive to integrate into other systems
  - Difficult to generate consistent reports over time and across projects
  - No data stewardship and change control
  - No "enrichment" processes
- **Clear**
  - Precise definitions for each data element and code values
  - Multiple sources continuously harmonized into canonical forms
  - Low cost to integrate and share with other systems
  - Designed for multiple purposes
  - Strong data stewardship and robust change control
  - Continual data enrichment

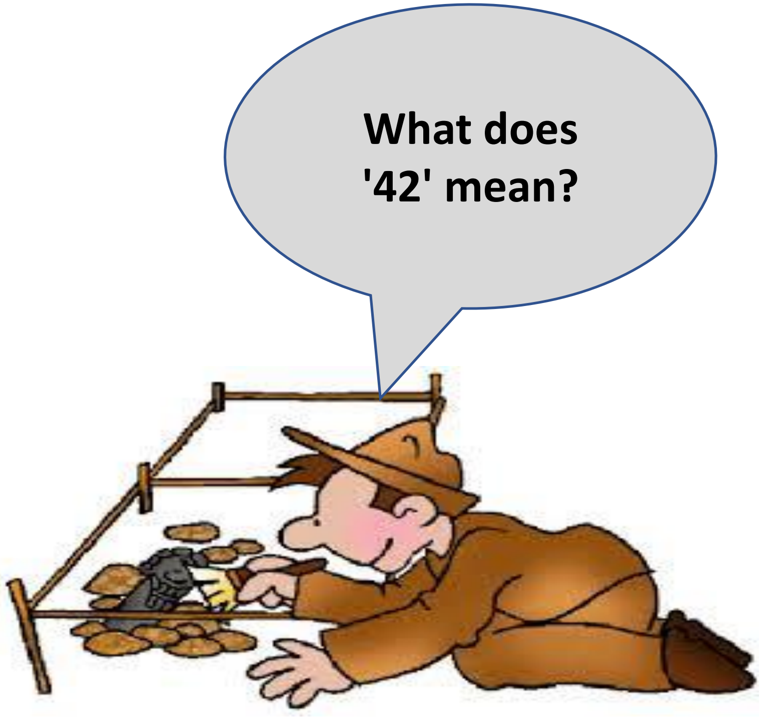
How clear are the semantics of our integration hub?

Data Analyzer / Data Profiler



midwest
architecture community
collaboration

Avoid Data Archaeology



- Time consuming task of converting numeric representations to symbolic representations
- Focus on strong metadata management and metadata services

XML Schemas (data quality)

Metadata and Reference Data (Semantics)

midwest
architecture community
collaboration

"Semantics" vs. "semantics"

- **Semantics** with an uppercase "S" usually refers to the Semantic Web technology stack (RDF, SPARQL, inference)
- **semantics** (with a lower case) usually refers to the process of creating shared meaning across multiple business units. This refers to the processes of Data Stewardship, Data Governance and metadata registries
- Our recommendation is to use 90% documents and RDF (Semantics) to link documents
- We do not recommend storing canonical documents in pure RDF

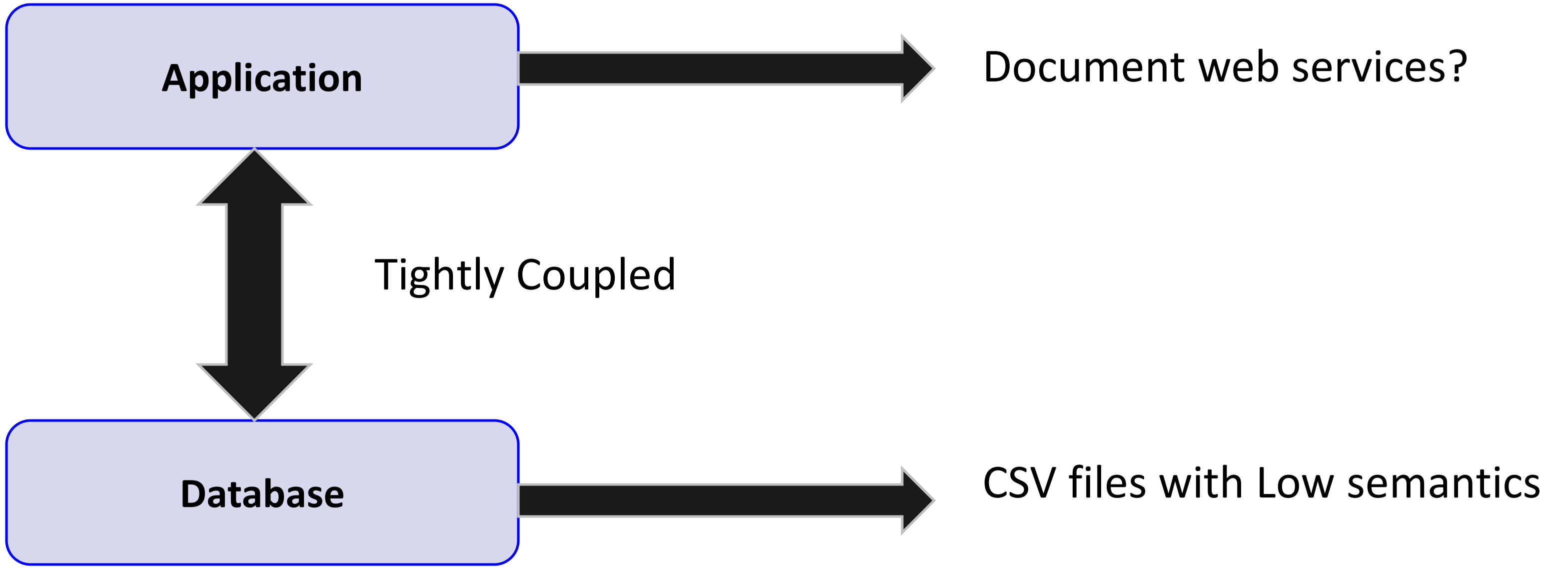
midwest
architecture community
collaboration

The Role of Search

document term weight	query term weight
$f_{t,d} \cdot \log \frac{N}{n_t}$	$\left(0.5 + 0.5 \frac{f_{t,q}}{\max_t f_{t,q}}\right) \cdot \log \frac{N}{n_t}$
$1 + \log f_{t,d}$	$\log(1 + \frac{N}{n_t})$
$(1 + \log f_{t,d}) \cdot \log \frac{N}{n_t}$	$(1 + \log f_{t,q}) \cdot \log \frac{N}{n_t}$

- Many systems don't integrate search well into their data
- Calculating keyword density is hard
- Calculating concept density is even harder
- Great data hubs must come up with ways to make it easy to find the data you need

Applications tightly coupled to database



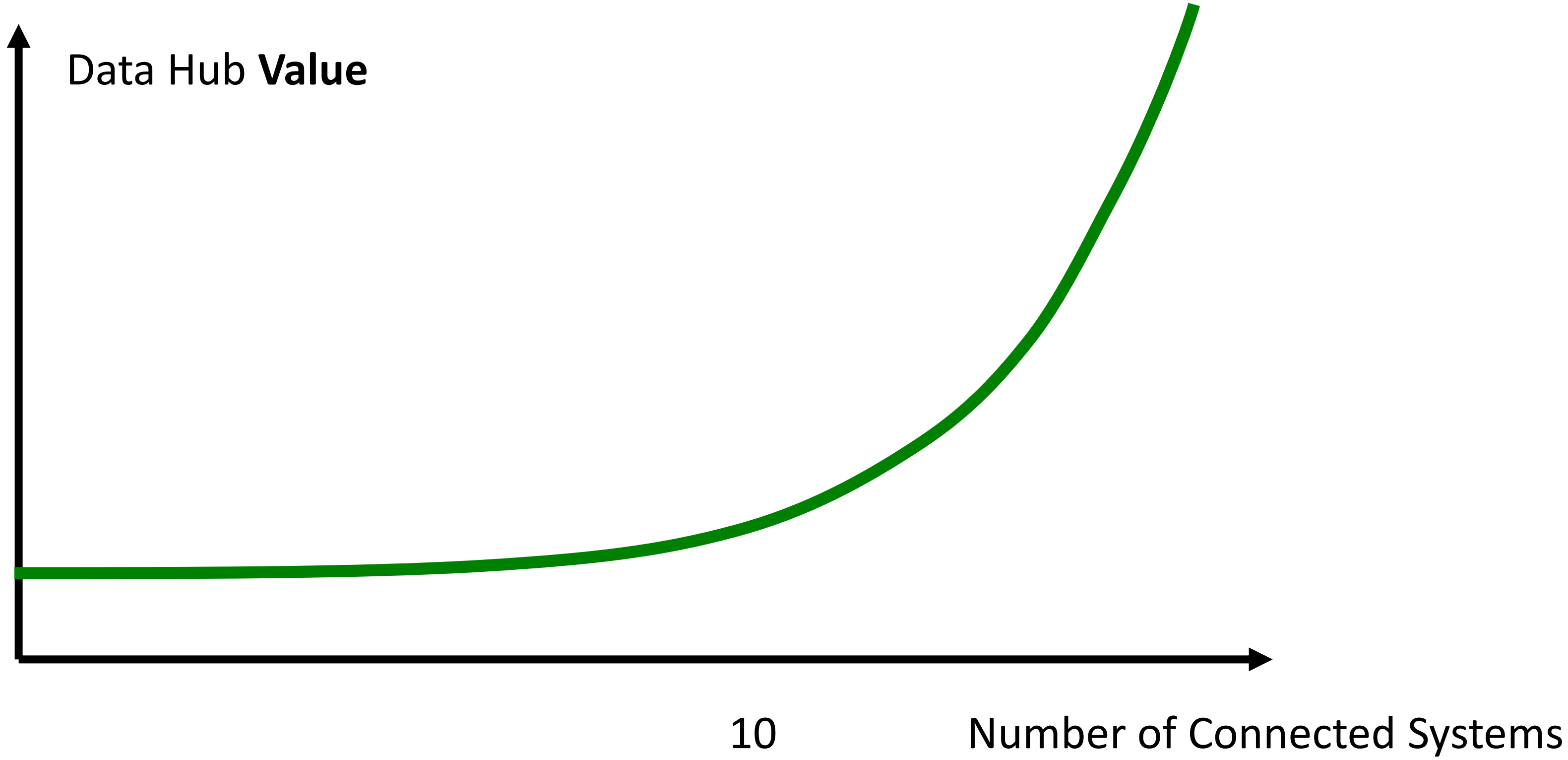
The root cause of many integration problems is that the application is tightly coupled to the database. Raw data dumped about of a typical RDBMS is almost useless without using the logic within and application.

midwest
architecture community
collaboration

Metcalf's Law

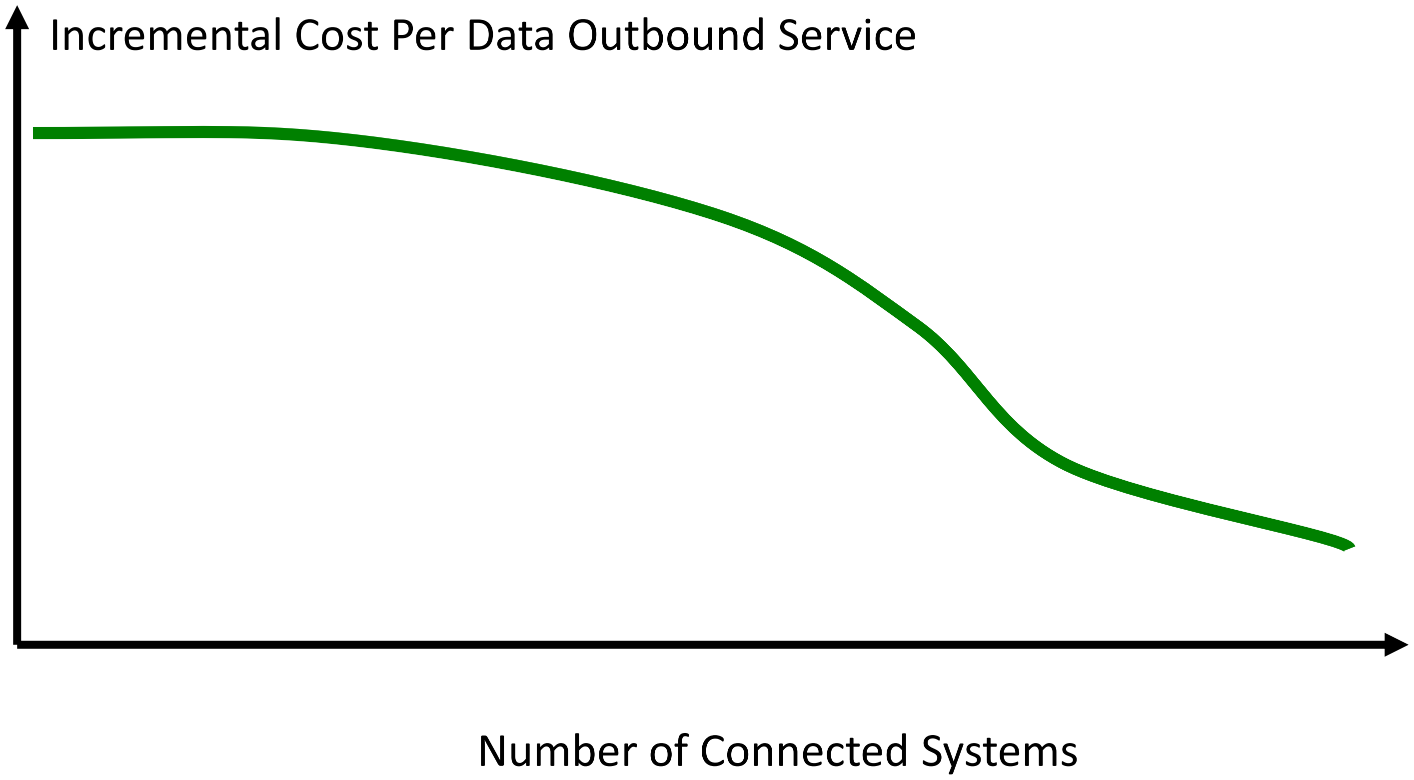
The value of a [system] is proportional to the square of the number of connected [systems].

https://en.wikipedia.org/wiki/Metcalfe%27s_law



midwest architecture community collaboration

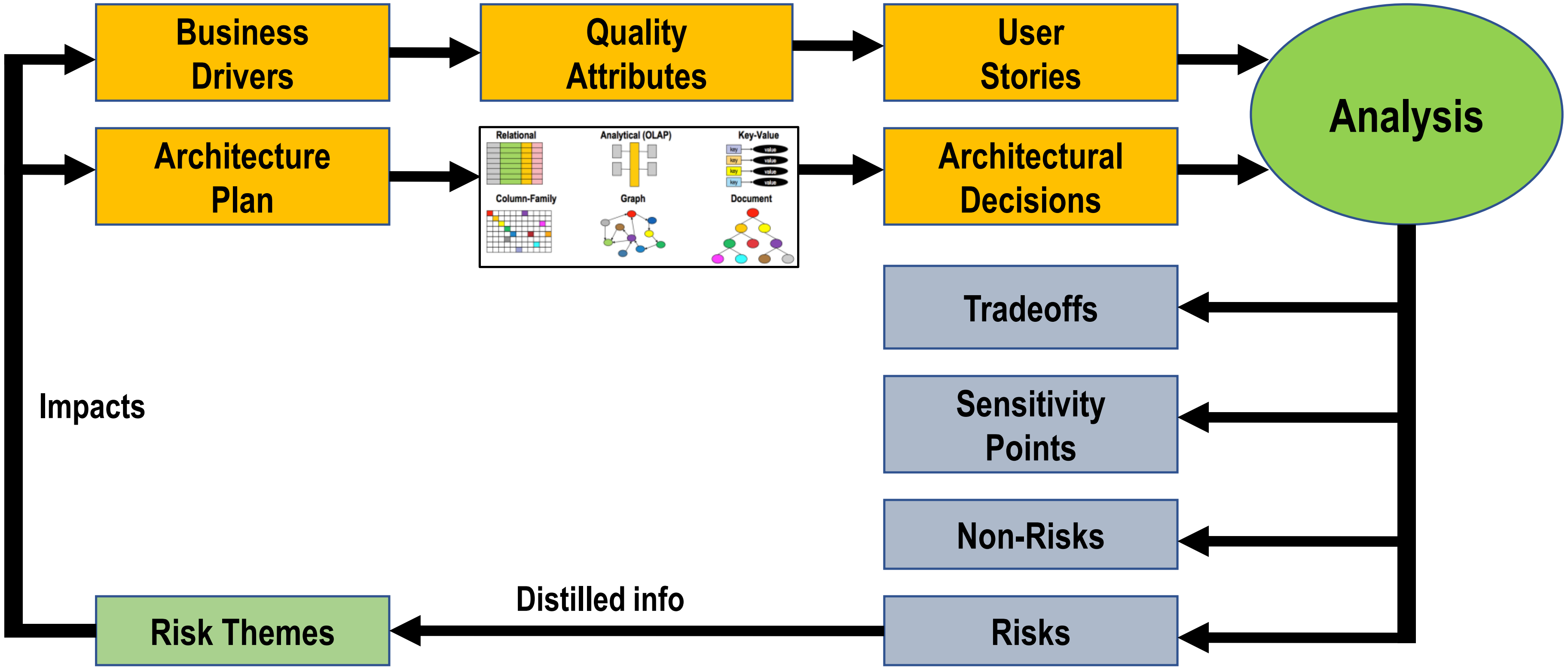
Lower Incremental Costs



Each new outbound data service can leverage prior data loaded in the data hub.

midwest
architecture community
collaboration

Architecture Tradeoff and Analysis Method (ATAM)



See Chapter 12 of *Making Sense of NoSQL*

midwest architecture community collaboration

Quality Attribute Tree

- How important
- How difficult in any given architecture

Scalability	Scales to many nodes	(H,L)	Ability store data on many nodes
	Distributed query	(H,L)	Move queries to data, not data to queries
Availability	Fault tolerant	(H,L)	Ability store data on many nodes
	Autosharding	(H,L)	Data automatically migrates to new nodes
Findability	Fulltext Search	(H,L)	Ability to perform serch on natural language text
	Customizable Ranking	(H,L)	Ability to change search ranking based on context
Schemaless	High variability	(H,L)	Easy to import high variability data
	No modeling	(H,L)	Easy to import data before modeling
Queryability	Expressivity	(H,L)	Query any attribute of data
	Correct query language	(H,L)	Query language fits the problem space
Transformability	Ease of transformation	(H,M)	Easy to write transforms
	Transform scalability	(H,L)	Transforms distributed over many servers
Affordability	Open Source License	(H,L)	No license fees
	Standards Based	(M,H)	Based on standards to reuse skills and tools
Interoperability	Integrates with reporting	(M,H)	Works with standardized reporting systems
	Standards Based	(M,H)	Based on standards for ease of portability
Security	Role based access control	(H,H)	Secure any document based on role
	Audit	(H,H)	Detailed audit reporting

midwest architecture community collaboration

Summary

- Deep Learning needs **lots** of data – typically millions of records
- Both Data Lakes and Data Hubs are great examples of **distributed** computing
- Both have **lower cost/TB/year** than RDBMS and are far more **flexible** than an RDBMS
- Be cautious about doing integration on a RDBMS unless you know you have homogeneous today and forever
- Use Data Lakes for ways to store log files or some forms of **simple** tabular data
- Use document and graph stores for building integration Data Hubs
- Use Data Hubs to power your Deep Learning models

Further Reading and Questions

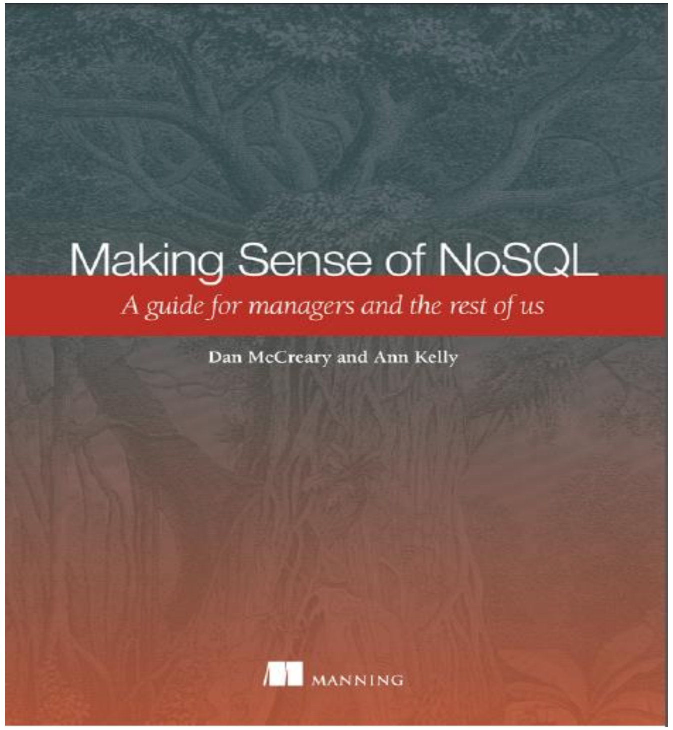
Thank You!

Dan McCreary

dan.mccreary@optum.com

twitter @dmccreary

LinkedIn http://www.linkedin.com/in/danmccreary



http://manning.com/mccreary

midwest architecture community collaboration